Timo Keränen

IOT-KÄYTTÖJÄRJESTELMIEN VERTAILU

IOT-KÄYTTÖJÄRJESTELMIEN VERTAILU

Timo Keränen
Opinnäytetyö
Kevät 2017
Tietotekniikan koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan koulutusohjelma

Tekijä: Timo Keränen
Opinnäytetyön nimi: IoT-käyttöjärjestelmien vertailu
Työn ohjaaja: Teemu Korpela
Työn valmistumislukukausi ja -vuosi: Kevät 2017
Sivumäärä: 28

Opinnäytetyön aiheena oli IoT-käyttöjärjestelmien vertailu, jossa tehtiin katsaus IoT-käyttöjärjestelmien erityispiirteisiin, yleisimpiin tällä hetkellä käytössä oleviin IoT-käyttöjärjestelmiin, IoT-laitteiden ja pilvipalvelujen rajapintoihin sekä muutamiin IoT-laitteiden tuottamaa avointa dataa tarjoamiin palveluihin.

Työ eteni tutkimalla ensin yleisesti käyttöjärjestelmiä ja niiden toimintaperiaatteita, sitten IoT-käyttöjärjestelmien erityispiirteitä. Tämän jälkeen tutkittiin yleisimpien IoT-käyttöjärjestelmien historiaa, ominaisuuksia ja toimintaa. Lopuksi tehtiin katsaus IoT-käyttöjärjestelmien ja pilvipalvelujen yhteenliittymiseen ja IoT-laitteiden tuottamaa dataa tarjoaviin avoimen datan palveluihin.

Yhteenvetona voidaan todeta, että IoT-käyttöjärjestelmiä on käytössä lukuisia ja ne eroavat toisistaan ohjelmointitavan, ohjelmointikielen sekä laite- ja muistivaatimusten osalta. IoT-laitteiden määrä maailmassa lisääntyy koko ajan ja yhä useampi niistä tuottaa dataa suoraan pilvipalveluihin.

Asiasanat: IoT, esineiden internet, käyttöjärjestelmät, pilvipalvelut, avoin data

ABSTRACT

Oulu University of Applied Sciences
Information technology

Author: Timo Keränen
Title of thesis: The comparison of IoT operating systems
Supervisor: Teemu Korpela
Term and year when the thesis was submitted: Spring 2017
Pages: 28

The subject of the thesis was to compare IoT operating systems, to study what are their special features compared to other operating systems, to study some of the most commonly used IoT operating systems, to study the connectivity between IoT devices and cloud services and to check some of the open data services which serve the data that IoT devices produce.

The work started with the common study of the operating systems and their operating principles. Then the special features of the IoT operating systems was studied and how they differ from the other operating systems. After that the study of the different IoT operating systems currently in use at the world was done. Then the connectivity between IoT devices and cloud services was studied, before checking some of the open data services which serve the data that IoT devices produce.

As a conclusion can be said that several different IoT operating systems are used around the world. They differ from each other by how they are programmed, what programming languages are used, how much they use memory- and power capacity and what devices they are meant for. Also the amount of IoT devices in the world increases all the time and increasing number of them produce data directly to the cloud services.

Keywords: IoT, Internet of things, operating systems, cloud services, open data

ALKULAUSE

Suurkiitokset vaimolleni tuesta pitkään kestäneessä opinnäytetyön tekemisessä. Kiitokset myös työn ohjaajalle Teemu Korpelalle sekä kieliasun tarkastajalle Tuula Hopeavuorelle.

Oulussa 27.5.2017

Timo Keränen

SISÄLLYS

1 JOHDANTO

Internet of Things (IoT) eli esineiden internet on nykyään yhä enemmän läsnä jokapäiväisessä elämässämme. Esineiden internetillä tarkoitetaan älyn lisäämistä laitteisiin tai tuotteisiin. Älyn avulla ne saadaan tuottamaan tietoa ympäristöstä ja toimittamaan tiedon eteenpäin tai tekemään jotain tiedon perusteella.

Jotta tätä älyä päästäisiin hyödyntämään, tarvitaan jokaiselle IoT-laitteelle käyttöjärjestelmä. Ilman kunnollista käyttöjärjestelmää laitteiden potentiaalia ei päästä hyödyntämään. Käyttöjärjestelmän tehokkuus korostuu etenkin IoT-laitteissa, joissa ei ole resursseja hukattavaksi asti kuten nykyajan pöytätietokoneissa. Jos käyttöjärjestelmä ei toimi tarpeeksi tehokkaasti, sensorilta tuleva viesti voi jäädä käsittelemättä ja IoT-laitteen toiminta käyttötarkoituksessaan heikkenee.

Tämän työn tarkoituksena on tarkastella IoT-käyttöjärjestelmiä, niiden erikoispiirteitä ja niiden eroja muista käyttöjärjestelmistä. Tämän lisäksi tarkastellaan suosituimpia IoT-käyttöjärjestelmiä ja niiden ominaisuuksia. Lopuksi tehdään katsaus IoT-käyttöjärjestelmien pilvirajapintoihin sekä avoimen datan palveluihin. (1.)

2 IOT-KÄYTTÖJÄRJESTELMÄT

Käyttöjärjestelmä on keskeinen tietokoneen ohjelmisto, joka mahdollistaa muiden ohjelmien toiminnan hallinnoimalla tietokoneen resursseja, ovat ne sitten fyysisiä (muisti, tallennustila, verkkoyhteys yms.) tai ohjelmallisia (muiden ohjelmien suorittaminen prosesseina, porttien tarjoaminen eri verkkoyhteyksille jne.). Käyttöjärjestelmä koostuu karkealla tasolla ytimestä, järjestelmäohjelmistosta ja komentotulkista. (2.)

Ydin (kernel) on käyttöjärjestelmän sydän, kuten nimikin sanoo. Ilman sitä käyttöjärjestelmä olisi hyödytön. Se kontrolloi sekä fyysisiä että ohjelmallisia resursseja. Ydin myös sallii muiden ohjelmistojen ja käyttäjien suorittaa erilaisia toimintoja. (2.)

Järjestelmäohjelmisto tukee käyttöjärjestelmän toimintoja ja tarjoaa ytimelle tarpeelliset ohjelmat, kuten debuggerit ja kääntäjät. Se myös tarjoaa käyttäjille pääsyn perusohjelmistoihin, kuten tekstinkäsittelyyn, taulukkolaskentaan yms. Komentotulkki tarjoaa käyttäjille käyttöliittymän ytimeen. (2.)

2.1 IoT-käyttöjärjestelmien erityispiirteet

Normaalit käyttöjärjestelmät sisältävät ominaisuuksia, joita IoT-verkoissa ei tarvita. Sulautetuissa järjestelmissä ajettava koodi on paljon suppeampaa ja yhtenäisempää, kuin tavallisessa käytössä olevissa järjestelmissä. IoT-verkkojen käyttöjärjestelmä pitää suunnitella sen erityiskäyttöön sopivaksi; energiankulutukseen täytyy kiinnittää erityistä huomiota, asynkronisesti syötettyä tietoa pitää pystyä käsittelemään sujuvasti. Tähän tarvitaan erityinen käyttöjärjestelmä, jonka täytyy myös toimia hyvinkin pienellä muistimäärällä. (3.)

2.1.1 Ohjelmointi

Käyttöjärjestelmässä pitää olla hyvä tuki samanaikaisuudelle. Tietoa voi tulla useasta eri lähteestä lähes samaan aikaan, joten laskentaa täytyy pystyä tekemään samanaikaisesti. (3.)

2.1.2 Sekvenssiohjelmointi

Sekvenssiohjelmoinnilla tarkoitetaan tehtävät peräkkäin suorittavaa järjestelmää. Siinä esimerkiksi pollataan ensin anturia, onko sillä uutta tietoa, ja jos on, käsitellään se heti. Sitten pollataan esimerkiksi vastaanotinta ja käsitellään sieltä tuleva tieto heti. Tämä ohjelmointitapa on yksinkertainen, mutta siinä on suuri riski menettää tietoa, jos uutta tietoa tulee samalla kun vanhaa vasta käsitellään. Riski kasvaa, jos tiedon käsittely kestää kauan eikä käytössä ole mitään puskuria tiedon välivarastointiin. (3.)

2.1.3 Prosessipohjainen ohjelmointi

Normaalit käyttöjärjestelmät tukevat monen prosessin yhtäaikaista suorittamista yhdellä prosessorilla. Tämä toimintatapa kuulostaa juuri sopivalta IoT-verkkoihin, mutta se ei sitä kuitenkaan ole. Suoritettavan prosessin vaihtaminen nimittäin kuluttaa paljon resursseja. Tämä korostuu etenkin, kun suoritettavat prosessit ovat pieniä, kuten IoT-verkoissa usein on. Tällöin varsinaisen tiedon käsittelyyn kuluva prosessoriaika jää pieneksi prosessien hallinnoimiseen verrattuna. Tästä syystä prosessipohjainen ohjelmointi ei sovellu hyvin IoT-verkkoihin. (3.)

2.1.4 Tapahtumapohjainen ohjelmointi

IoT-verkkojen reaktiivisen luonteen takia tapahtumapohjainen ohjelmointi soveltuu niihin hyvin. Siinä järjestelmä odottaa jotain tapahtumaa – oli se sitten tiedon saapuminen anturilta, paketin saapuminen vastaanottimelta tai laskurin umpeutuminen. Tällainen tapahtuma käsitellään nopeasti ja yksinkertaisesti, siinä ainoastaan tallennetaan tapahtuman sisältämä tieto sekä se, mikä tapahtuma oli kyseessä. Varsinainen tiedon käsittely tapahtuu tästä tapahtumakäsittelijästä erillään. Tällaiset tapahtumakäsittelijät keskeyttävät muun koodin ajon, mutta koska ne ovat kooltaan pieniä ja yksinkertaisia, ne eivät huomattavasti häiritse muun koodin ajoa. Tapahtumakäsittelijät eivät voi keskeyttää toistensa ajoa (koska se johtaisi monimutkaiseen pinonhallintaan), vaan ne suoritetaan toinen toisensa jälkeen. Tapahtumapohjainen ohjelmointi tuo etuja sekvenssi- ja prosessipohjaiseen ohjelmontiin verrattuna. Samalla laitteistolla suoritetussa vertailussa tapahtumapohjainen ohjelmointi paransi suorituskykyä, muistintarvetta sekä virrankulutusta jopa 30-kertaisesti. (3; 4.)

2.2 Käyttöjärjestelmän rakenne ja protokollapino

Perinteinen kommunikaatioprotokollan rakenne on kerroksellinen. Yksittäiset protokollat on pinottu toistensa päälle siten, että jokainen kerros voi käyttää vain suoraan alapuolellaan olevia funktioita. Tätä rakennetta käyttäen prokollapino pysyy hallittavana, helppokäyttöisenä ja modulaarisena. IoT-verkkojen käytössä tällainen tiukka kerroksellinen rakenne ei useinkaan riitä. (3.)

Kun kerroksellista rakennetta pilkotaan pienempiin osiin, jotka voivat viestiä keskenään kerrossijainnista riippumatta, puhutaan komponenttirakenteesta. Tällaiset komponentit toteuttavat usein hyvin pienen, tarkkaan määritellyn tehtävän ja viestivät keskenään selvästi määriteltyjen rajapintojen kautta. Tällainen komponenttimalli sopii hyvin tapahtumapohjaisiin IoT-verkkoihin. (3.)

3 KÄYTETYIMMÄT IOT-KÄYTTÖJÄRJESTELMÄT

3.1 TinyOS

TinyOS kehitettiin Kalifornian yliopistossa Berkeleyssä Yhdysvalloissa osana Yhdysvaltain puolustusministeriön DARPA NEST (Defense Advanced Research Projects Agency Neural Engineering, Science, and Technology) -ohjelmaa yhteistyössä Intelin tutkimusosaston kanssa. Ensimmäinen versio kehitettiin 1999 ja julkinen TinyOS:stä tuli 2000. Sittemmin siitä on tullut yksi suosituimmista IoT-käyttöjärjestelmistä. Viimeisin julkaistu versio on vuodelta 2012. (5.)

TinyOS on kirjoitettu NesC-ohjelmointikielellä. Siinä käytetään komponenttipohjaista arkkitehtuuria. Siinä yksinkertaisia funktioita sisällytetään komponentteihin, joissa on selkeät rajapinnat ja monimutkaisia funktioita voidaan implementoida yhdistelemällä komponentteja. Komponenttipohjainen arkkitehtuuri mahdollistaa usein tapahtuvat muutokset koodiin samalla, kun koodin koko pysyy mahdollisimman pienenä. Tapahtumapohjaisessa suoritusmallissa ei ole käyttäjän tai ytimen aiheuttamia rajoituksia, joten se tukee hyvin koodin rinnakkaista suorittamista. (6.)

3.2 FreeRTOS

FreeRTOS on suosittu reaaliaikakäyttöjärjestelmä sulautetuille laitteille. Se on suunniteltu pieneksi ja yksinkertaiseksi. Ydin itsessään koostuu ainoastaan kolmesta C-kielisestä tiedostosta. Koodin luettavuuden, portattavuuden ja ylläpidon vuoksi se on kirjoitettu enimmäkseen C-kielellä, mutta muutamia arkkitehtuuri-spesifisiä ajoitusrutiineja on kirjoitettu Assembler-kielellä. (7.)

FreeRTOS tarjoaa metodeja useisiin säikeisiin tai taskeihin, lippuihin ja ohjelmallisiin laskureihin. Ydin voidaan konfiguroida erityisen vähän virtaa kuluttavaan tick-less-tilaan, jossa ajastinkeskeytykset eivät tule säännöllisin välein vaan ainoastaan pyynnöstä. Säikeiden prioriteetit ovat tuettuna. FreeRTOS-sovellukset voidaan sijoittaa muistiin täysin staattisesti tai ne voidaan sijoittaa dynaamisesti viiden eri muistiallokaation mukaisesti. (7.)

Mitään yleensä käyttöjärjestelmissä olevista monimutkaisista ominaisuuksista, laiteajureita, kehittynyttä muistinhallintaa, käyttäjätilejä tai verkonhallintaa ei ole FreeRTOS:ssa, vaan siinä paino on pienuudessa ja suorituksen nopeudessa. FreeRTOS:n voidaan ajatella olevan pikemminkin "säiekirjasto" kuin "käyttöjärjestelmä", vaikka komentorivitulkki ja I/O-käyttöliittymä ovatkin saatavilla lisäosina. (7.)

FreeRTOS toteuttaa useat säikeet tavalla, jossa isäntäohjelma kutsuu säiettä säännöllisin, lyhyin aikavälein. Säiettä vaihdetaan prioriteetin ja round-robin-aikataulutuksen mukaisesti. Yleinen aikaväli on sadas- tai tuhannesosasekunti laiteajastimen keskeyttämänä, mutta tätä aikaväliä usein muutetaan tarvittavan sovelluksen mukaisesti. (7.)

FreeRTOS tarjoaa valmiita konfiguraatioita ja demoja jokaiselle tuetulle porttaukselle ja kääntäjälle ja on nopea sovelluskehitysympäristö. FreeRTOS.org-sivusto sisältää myös paljon dokumentaatiota, RTOS-tutoriaaleja ja RTOS-suunnittelumalleja. (7.)

3.3 µC/OS II & III

Micro-Controller Operating Systems (MicroC/OS tai µC/OS) on reaaliaikakäyttöjärjestelmä, jonka kehitti alun perin Jean J. Labrosse 1991. Se julkaistiin alun perin kolmiosaisena artikkelina Embedded Systems Programming -lehdessä ja uC/OS The Real-Time Kernel -kirjassa. Se on prioriteettipohjainen keskeytettyyn moniajoon pohjautuva reaaliaikaydin mikroprosessoreille, kirjoitettu enimmäkseen C-kielellä. (8.)

MicroC/OS:ssä voidaan useita funktoita määritellä C-kielellä ja jokainen funktio voidaan ajaa erillisenä säikeenä tai taskina. Jokainen taski ajetaan eri prioriteetillä ja sitä suoritetaan kuten CPU:n omaa taskia. Korkean prioriteetin taskit voivat keskeyttää alemman prioriteetin taskin suorituksen milloin vain. Korkeamman prioriteetin taskit käyttävät käyttöjärjestelmän palveluita antamaan alemman prioriteetin taskeille suoritusaikaa. Käyttöjärjestelmä tarjoaa palveluita taskien hallintaan, taskien väliseen kommunikointiin, muistinhallintaan ja ajoitukseen. Taskit kirjoitetaan joko ikuiseen silmukkaan tai taskiksi, joka tuhoaa itsensä ajon jälkeen. (8.)

µC/OS-II

Alkuperäisen MicroC/OS:n lähdekoodiin perustuen MicroC/OS-II esiteltiin kaupallisena tuotteena 1998. Se voi käsitellä maksimissaan 255 applikaatiotaskia ja sen kokoa voidaan skaalata 5–24 kilotavun välillä riippuen tarvittavista ominaisuuksista. (8.)

Suurin osa MicroC/OS-II:sta on kirjoitettu ANSI-C:llä, joten se voidaan helposti ottaa käyttöön monissa eri laitteissa. MicroC/OS-II on suunniteltu sulautettuihin laitteisiin. Sitä käytetään mm. lentokoneissa, lääketieteellisissä laitteissa, tietoliikennelaitteissa, matkapuhelimissa, autoteollisuudessa jne. (8.)

MicroC/OS-II on moniajokäyttöjärjestelmä. Jokaisella taskilla on ikuinen silmukka ja taski voi olla viidessä eri tilassa:

- lepäävänä
- valmiina
- ajossa
- odottamassa (tapahtumaa)
- keskeytettynä.

MicroC/OS-II voi käsitellä 255 erillistä taskia, joista tosin kahdeksan on varattu järjestelmän käyttöön, joten käyttäjälle jää 247 taskia. (8.)

µC/OS-III

MicroC/OS-III esiteltiin vuonna 2009. Siitä löytyvät kaikki kakkosversion ominaisuudet. Suurimpana erona kakkosversioon on mahdollisten taskien lukumäärä. Kakkosversiossa voi olla ainoastaan yksi taski jokaista 255 prioriteettitasoa kohti eli yhteensä 255 taskia. Kolmosversiossa taskien ja prioriteettitasojen määrää rajoittaa ainoastaan käytettävän prosessorin kyky hallinnoida muistia.

MicroC/OS-II:n ja MicroC/OS-III:n kehityksestä vastaa nykyään Micrium Inc. -yhtiö joka hallinnoi niiden käyttöä. (8.)

3.4 Contiki

Contiki on avoimeen lähdekoodiin perustuva, erittäin helposti erilaisiin laitteisiin sovellettava, keskeyttävää moniajoa tukeva käyttöjärjestelmä, jonka ensimmäisen version kehitti Adam Dunkels Swedish Institute of Computer Sciencessä Ruotsissa 2002. (9.)

Contiki on ensimmäinen IoT-käyttöjärjestelmä joka toteuttaa TCP/IP-rajapinnan, mukaanlukien IPv6:n. Contiki tukee myös uusimpia matalan tehonkulutuksen langattomia standardeja, kuten 6lowpan, RPL ja CoAP. Contikilla ohjelmistokehitys on helppoa ja nopeaa – kaikki sovellukset tehdään C-kielellä. Contiki on suunniteltu pieniin, jopa vain muutaman kilotavun keskusmuistin järjestelmiin. Contikin muistinhallinta on siis erittäin tehokasta. Contiki tukee myös dynaamista, ajonaikaista moduulien lataamista ja linkitystä. Tämä mahdollistaa sovellusten toiminnan muuttamisen niiden käynnistämisen jälkeen ilman IoT-laitteen uudelleenkäynnistystä. (10.)

Vuonna 2012 Contiki-kehittäjät perustivat yrityksen nimeltä Thingsquare viedäkseen Contiki-käyttöjärjestelmän osaksi pilvipalveluita. (10.)

3.5 Microsoft .NET Micro

.NET Micro Framework (NETMF) on Microsoftin kehittämä .NET-pohjainen alusta laitteille, joissa on vähintään 256 kB flash-muistia ja 64 kB RAM-muistia. Se sisältää pienen version .NET Common Language Runtimestä ja sille voidaan tehdä sovelluksia C#:lla ja Visual Basic .NET:llä. Sovelluksista voidaan myös etsiä ja poistaa virheitä Microsoft Visual Studion avulla. (11.)

Muihin .NET-ympäristöihin verratuna NETMF:ssä on eroavaisuutena mm. seuraavat:

- Sitä voidaan ajaa ilman muuta käyttöjärjestelmää tai toisen käyttöjärjestelmän päällä.
- Se tukee yleisiä sulautettuja lisälaitteita ja yhteysvälineitä, mm. flash-muisti, EEPROM, GPIO, sarjaportti ja USB.
- Se on optimoitu energiaasäästäviin akkukäyttöisiin laitteisiin.
- Siinä ei ole tarvetta muistinkäsittely-yksikölle.

- Siinä on monisäietuki jopa ajettaessa yksisäikeisessä käyttöjärjeslmässä.
- Se tukee porttausta muihin arkkitehtuureihin.
- Sen ajureita monille laitteille voidaan kirjoittaa C#:lla.
- Siinä laitteiden jumiintumiset ja kaatumiset saadaan kiinni ajonaikaisilla rajoituksilla. (11.)

3.6 Huawei LiteOS

Huawei LiteOS on kiinalaisen tietoliikennejätin vuonna 2015 julkistama ohjelmistoalusta, joka yhdistää IoT-käyttöjärjestelmän ja väliohjelmiston. Se on erittäin kevyt ja energiatehokas, sillä ytimen koko on vain 10 kB ja tavallisella AA-paristolla se voi pyöriä viisi vuotta. Huawei LiteOS käynnistyy erittäin nopeasti, moduulit ladataan kerros kerrokselta erityisellä hajalataus-algoritmillä. IoT-laitteiden väliseen kommunikointiin Huawei LiteOS tarjoaa niin pitkän kantaman LTE- ja NB-IoT-yhteydet kuin lyhyen kantaman WiFi- ja 6LoWPAN-yhteydet. Huawei on rakentanut käyttöjärjestelmänsä ympärille ekosysteemin, joten se tarjoaa tukea mahdollisimman monenlaiselle CPU-arkkitehtuurille, mukaan lukien ARM- ja x86-laitteet. Yksi Huawein yhteistyökumppaneista LiteOS:n kehityksessä on suomalainen Kone. (12.)

3.7 Android Things

Android Things (koodinimeltään Brillo) on Android-pohjainen sulautettujen laitteiden käyttöjärjestelmä, jonka Google julkisti 2015 tekevänsä. Se on suunnattu eri valmistajien vähämuistisiin ja matalan virrankulutuksen IoT-laitteisiin. Se tukee matalavirrankulutuksista Bluetooth Low Energy -teknologiaa ja langattomia WiFi-yhteyksiä. (13).

Google myös julkisti Weave-protokollan, joka mahdollistaa IoT-laitteiden keskinäisen kommunikoinnin. (13).

Android Things -käyttöjärjestelmää ei vielä ole virallisesti julkistettu, mutta sen on sanottu parantavan IoT-laitteiden lähes olematonta tietoturvaa Googlen automaattisilla ohjelmistopäivityksillä. (14.)

3.8 Nano-RK

Nano-RK on Carnegie Mellon -yliopistossa 2005 kehitetty reaaliaikakäyttöjärjestelmä mikrokontrollereille käytettäväksi IoT-verkoissa. Nano-RK tukee kiinteäprioriteettistä täysin keskeyttävää moniajoa. Nano-RK:n nimessä oleva Nano vihjaa pienuuteen ja Nano-RK onkin pieni - se kuluttaa vain 2 kB RAM-muistia ja 18 kB ROM-muistia. (15.)

Nano-RK:n nimessä oleva RK on lyhenne sanoista Resource Kernel. Sillä tarkoitetaan käyttöjärjestelmän ydintä, joka resursseja voidaan varata käyttöön vain rajoitetusti. Taskille voidan esimerkiksi antaa ajoaikaa vain 10 ms joka 150 ms:n välein (CPU-varaus) tai IoT-laitteen voidaan sallia vain lähettävän 10 pakettia minuutissa (verkkovaraus). (15.)

Nano-RK on avoimeen lähdekoodiin perustuva, kirjoitettu C-ohjelmointikielellä ja sitä voidaan ajaa Atmel-pohjaisissa mikrokontrollereissa. (15.)

3.9 RIOT OS

RIOT on pieni käyttöjärjestelmä verkossa oleville IoT-laitteille, joissa on vähän muistia ja jotka kuluttavat vähän virtaa. Se on avoimeen lähdekoodiin perustuva ja julkaistu LGPL-lisenssin alla, mistä syystä sitä pidetäänkin IoT-Linuxina. (16.)

RIOT kehitettiin alunperin saksalaisten ja ranskalaisten yliopistojen yhteistyönä, mutta nykyään sitä kehitetään kansanvälisen yhteisön toimesta. RIOT perustuu mikrokernel-arkkitehtuurille. Erona muihin vastaaviin vähän muistia kuluttaviin IoT-käyttöjärjestelmiin RIOT-ohjelmia voidaan ohjelmoida C- ja C++ -kielillä ja se tukee täyttä monisäikeisyyttä ja reaaliaikaominaisuuksia. (16.)

3.10 Windows 10 IoT Core

Windows 10 IoT Core on Microsoftin vuonna 2015 lanseeraama versio Windows 10 -käyttöjärjestelmästä, joka on suunnattu pienille laitteille, ilman näyttöä tai näytön kanssa. Sitä voidaan käyttää sekä ARM- että x86/x64-laitteissa. (17.)

Windows 10 IoT Core -sovellusten tekemiseen voidaan käyttää Microsoft Visual Studio -ohjelmistokehitystyökalua. Tämä mahdollistaa tehokkaan

ohjelmistokehityksen suoraan IoT-laitteille PC-maailmasta tutulla työkalulla, kehittyneen debuggauksen ja eri ohjelmointikielien käyttämisen. Visual Studiosta on saatavilla Community-versio, joka on yksityis- ja opetuskäytössä ilmainen. (17.)

Windows 10 IoT Core tukee myös Universal Windows Platform (UWP) -arkkitehtuuria. Tämä mahdollistaa Windows-universaalien sovellusten tekemisen, jotka toimivat useilla eri Windows-laitteilla ilman sovelluksen laitekohtaista räätälöintiä. (17.)

3.11 mbed OS

mbed OS on puolijohdevalmistaja ARM:n yhteistyökumppaneineen kehittämä avoimen lähdekoodin IoT-käyttöjärjestelmä. Sen ensimmäinen versio julkaistiin 2009 ja kuten arvata saattaa, se on suunnattu ARM:n omiin mikrokontrollereihin perustuviin IoT-laitteisiin. (18.)

Sovelluksia mbed OS:lle voi tehdä C/C++ -kielellä pilvipalvelussa olevalla online-kehitystyökalulla, eli mitään ei tarvitse asentaa omalle tietokoneelle. Täten ohjelmistokehitys voi alkaa todella nopeasti, ilman erillisten kehitystyökalujen asentamista ja asetusten säätämistä. (18.)

mbed OS tukee monisäikeistä reaaliaikaista ohjelmistokehitystä. Se myös tukee monia kommunikaatiostandardeja, kuten WiFi, Ethernet, Bluetooth LE, NFC ja 6LoWPAN. (18.)

3.12 Linux

IoT-laitteiden käyttöjärjestelmänä voidaan käyttää myös Linuxia. Linux tarjoaa vakaan, kehittäjäystävällisen käyttöjärjestelmän, joka täyttää monia IoT-käyttöjärjestelmälle sopivia piirteitä:

- Skaalautuvuus, sopii monenlaisille erilaisille laitteille.
- Modulaarinen, mukaan voidaan valita vain ne komponentit, joita tarvitaan.
- Datayhteyksien monipuolisuus, käytössä on monta eri standardia, esimerkiksi WiFi, Ethernet, USB, Bluetooth.

- Luotettavuus, Linux on osoittanut luotettavuutensa monissa turvallisuuskriittisissä sovelluksissa. (19.)

Linux ei kuitenkaan sovellu kaikkien IoT-laitteiden käyttöjärjestelmäksi. Vaikka sitä karsitaan reilulla kädellä, se on silti iso ohjelmisto IoT-laitteiden mittapuulla. Minimivaatimukset toimivalle Linux-järjestelmälle ovat 2–6 MB RAM-muistia ja 8–16 MB massamuistia. Tästä johtuen Linux soveltuukin parhaiten IoT-laitteisiin, joissa on iso graafinen käyttöliittymä. (20.)

On myös olemassa erityisesti sulautettuihin laitteisiin suuntautuneita Linux-julkaisuja, kuten avoimen lähdekoodin yhteistyöhanke Yocto Project sekä Ubuntu Core (21; 22).

4 IOT JA PILVIPALVELUT

4.1 IoT-käyttöjärjestelmien pilvirajapinnat

Jos 35 vuotta sitten esitelty kotitietokone Commodore 64 saadaan kytkettyä internetiin (23), voidaan olettaa, että mikä tahansa IoT-käyttöjärjestelmä voidaan liittää pilvipalveluun näkemällä tarpeeksi vaivaa. On kuitenkin käyttöjärjestelmiä, joissa pilvipalveluja on ajateltu valmiiksi. Näitä ovat esimerkiksi

- Windows 10 IoT Core ja Azure
- Huawei LiteOS ja OceanConnect
- Android Things ja Google Cloud.

4.1.1 Windows 10 IoT Core ja Azure

Windows 10 IoT Core -laitteet voidaan yhdistää Microsoftin Azure-pilvipalveluun käyttäen Microsoftin käynnistämää avoimen lähdekoodin projektia nimeltä Connect The Dots. Siinä käytetään ennalta määriteltyä JSON (JavaScript Object Notation) -tiedostomuotoa IoT-laitteiden lähettämän tiedon välitykseen suoraan Azure IoT Hub:lle. Laitteista suoraan pilveen voidaan lähettää esimerkiksi sensoridataa ja hälytyksiä sisältäviä viestejä, laitteen tilatietoja ja mediatiedostoja, kuten valvontakameran kuvia. Myös tiedonsiirto suoraan pilvestä laittesiin on mahdollista. Näin voidaan esimerkiksi käskeä laite kytkemään tuuletin päälle lämpötilan noustessa, muuttaa laitteen sensoridatan lähetysväliä ja lähettää yhdensuuntaisia huomioviestejä laitteille. (24; 25.)

4.1.2 Huawei LiteOS ja OceanConnect

Huawei LiteOS:n yhteysprotokollaan on sisäänrakennettu yhteys Huawei OceanConnectiin. Se on Huawein käynnistämä avoin ekosysteemi, tarkoitettu IoT-, pilvilaskenta- ja Big Data (massadata) -teknologioille. OceanConnectin ytimessä on IoT Connection Management Platform, jonka kautta tarjotaan yli 170 avointa rajapintaa hyödynnettäviksi. Se tarjoaa myös valmiita verkkosovelluksia helpottamaan laitteiden kytkemistä osaksi ekosysteemiä. Tällaisia sovelluksia ovat esimerkiksi älykoti ja auton verkkoon kytkevä Connected Car. OceanConnect tarjoaa myös Big Data -analyysiä ja reaaliaikaista laitteiden

ohjausta. Se myös tukee Huawein omien standardien lisäksi kaikkia yleisiä IoT-standardeja, kuten Z-Wave, ZigBee ja Bluetooth. (26.)

4.1.3 Android Things ja Google Cloud

Googlen Android Things -käyttöjärjestelmää käyttävät IoT-laitteet kytkeytyvät luontevasti saman yhtiön Google Cloud -pilvipalveluun. Yhteyden muodostaminen tapahtuu Google Weave -kommunikaatioalustaa käyttäen. Se sisältää Weave Device SDK -sovelluskehitysympäristön ja Weave Serverin, joita käyttäen myös muut laitevalmistajat voivat kytkeä laitteensa Google Cloud -palveluihin. Weave Server mahdollistaa laitteiden salatun rekisteröitymisen pilvipalveluun, komentojen välittämisen, laitteiden tilojen tallentamisen ja integraation Googlen muihin palveluihin. (27.)

Google Cloud tukee myös avoimen lähdekoodin viestien lähetysprotokollaa gRPC:tä, joka on erittäin tehokas IoT-telemetriatiedon lähettämisessä. Google Cloud tarjoaa myös tehokkaat työkalut IoT-laitteilta saatavan tiedon säilömiseen, analysoimiseen ja saattamiseen esitettävään muotoon. (28.)

4.2 IoT ja pilvipalveluiden hyödyntäminen avoimen datan muodossa

Erilaiset IoT-laitteet tuottavat paljon dataa, jota tallennetaan pilvipalveluihin. Osa tästä datasta jaetaan vapaasti ja maksutta kansalaisten ja yritysten hyödynnettäväksi. Tällaista dataa kutsutaan avoimeksi dataksi. Avoimen datan tulee täyttää seuraavat ehdot:

- Julkisuus, data ei saa sisältää tietoa, joka vaarantaa yksityisyydensuojan tai yleistä turvallisuutta.
- Tekninen saatavuus, data pitää olla helposti tietokoneohjelmistoilla käsiteltävässä muodossa.
- Maksuttomuus, dataa pitää voida käyttää maksutta, ilman mitään byrokratiaa.
- Uudelleenkäyttö, datan pitää olla uudelleenkäytettävissä selkeästi kerrotuilla ehdoilla. (29.)

Seuraavaksi muutamia esimerkkejä avoimesta datasta:

4.2.1 6Aika

6Aika – Avoimet ja älykkäät palvelut on Suomen kuuden suurimman kaupungin (Helsinki, Espoo, Tampere, Vantaa, Oulu, Turku) kestävän kehittämisen strategia, jonka yhtenä painopistealueena on avoin data ja rajapinnat. Siinä kaupungit avaavat julkisen datansa kansalaisten ja yritysten käyttöön harmonisoiden samalla palvelurajapintojaan. (30.) Tässä muutamia esimerkkejä:

Oulun liikennedata

Oulunliikenne.fi tarjoaa avoimia rajapintoja, joista nähdään autoliikenteen liikennemäärä ja nopeustiedot, tiesääasemien tiedot, kelikameroiden kuvat ja pysäköintitalojen käyttöastetiedot. Pyöräilijöistä ja kävelijöistä nähtävänä on määrät eri mittauspisteissä. Joukkoliikenteestä tarjolla on rajapintoja pysäkkiennusteista, bussien reaaliaikaisesta sijaintitiedosta ja häiriötiedotteista. (31.)

Digitraffic rautatieliikenne

Liikenneviraston ratakapasiteetin hallinnan LIIKE-järjestelmästä poimitaan suoraan tietoa avoimen datan Digitraffic-rajapintaan. Tämän rajapinnan kautta on saatavilla tietoa junien reaaliaikaisesta seurannasta, junien kokoonpanotiedoista, junaliikenteen historiatietoja ja junien aikataulutietoja. (32.)

Digitraffic tieliikenne

Liikenneviraston tieliikennetietoa on saatavilla monipuolisesti Digitraffic-rajapinnan kautta. Sieltä saadaan tietoa LAM (Liikenteen automaattinen mittausasema) -asemien mittauspisteistä, tiejaksojen keliennusteita, tiesääasemien mittaustietoja, häiriötiedotteita, tieasemien tilatietoja, tietoa nopeusrajoituksista, kelikameroiden kautta tietoa tienpinnan tilasta ja liikennetilanteesta sekä sujuvuustietoa eri tieosuuksilta. (33.)

Tampereen reaaliaikainen liikennevalodata

Tampereen kaupunki on avannut avointen rajapintojen kautta käytettäväksi reaaliaikaista tietoa liikennevaloliittymistä. Näistä liittymistä saadaan tietoa muun muassa liikennemääristä, liikennevalo-opastimien ja -ilmaisimien tiloista ja odotusajoista. Näitä tietoja on kohdennettu esimerkiksi yrityksille, jotka kehittävät älyliikennepalveluja tai analysoivat liikennetietoja. (34.)

4.2.2 Ilmatieteen laitoksen avoin data

Ilmatieteen laitoksen tietoaineistoista suurin osa on saatavilla avoimena datana. Tämä data voidaan jakaa reaaliaikaisiin tietoihin, aikasarjoihin sekä ennustetietoihin:

- Reaaliaikaisina havaintoina saadaan sääasemilta esimerkiksi tuuli-, lämpötila-, kosteus-, ilmanpaine-, sade-, merivedenkorkeus- ja allokkohavaintoja. Näiden lisäksi saadaan säätutkakuvia ja salamoiden paikannustietoja.
- Havaintojen aikasarjoja ovat ilmastohavainnot vuodesta 1959, meriveden korkeushavainnot vuodesta 1971 ja aaltohavainnot vuodesta 2005.
- Ennustetietoja ovat kansallisen säämallin tuorein ennuste, sisältäen mm. pintasäätietoja, merimallien tuoreimmat ennusteet ja ilmastonmuutosskenaariot tulevaisuuteen. (35.)

Ilmatieteen laitos on myös laittanut avointa dataa jakoon Amazonin pilvipalvelun kautta. Kahden vuoden mittaisessa pilottikokeilussa tavoitteena on luoda edellytyksiä uuden liiketoiminnan syntymiselle ja lisätä säädatan hyödynnettävyyttä ja tehokkuutta sen käyttämiseen. Alkuvaiheessa Amazon Web Service -pilvipalvelun kautta on saatavilla koko Euroopan kattava sääennustemalli Hirlam. Malli on saatavilla samalla sisällöllä ja samassa muodossa kuin Ilmatieteen laitoksen omassa avoimen datan palvelussa. Ilmatieteen laitoksen palvelusta poiketen Amazonissa on saatavilla myös vanhojen malliajojen data. (36.)

5 YHTEENVETO

Opinnäytetyön tavoitteena oli tarkastella IoT-käyttöjärjestelmiä, niiden erikoispiirteitä ja eroja muista käyttöjärjestelmistä. Tämän lisäksi tarkasteltiin suosituimpia IoT-käyttöjärjestelmiä ja niiden ominaisuuksia. Lopuksi tehtiin katsaus IoT-käyttöjärjestelmien pilvirajapintoihin sekä avoimen datan palveluihin.

IoT-käyttöjärjestelmiä tarkastellessa tuli selväksi, että niitä on saatavilla joka lähtöön, erilaisille laitealustoille ja muistikonfiguraatioille, kaupallisia ja ilmaisia, suurten valmistajien tekemiä sekä pienten yhteisöjen kehittämiä. Mitään yhden tai muutaman IoT-käyttöjärjestelmän dominointia ei ole havaittavissa, kuten voidaan sanoa tilanteen olevan matkapuhelimissa tai pöytätietokoneissa. En myöskään näe tilanteen muuttuvan lähitulevaisuudessa.

Avoimen datan hyödyntämissä tuntuu, kuin olisimme vasta polun alussa. Erilaiset organisaatiot ovat lähteneet kiitettävästi julkaisemaan dataansa avoimesti hyödynnettäväksi ja tulemme varmasti näkemään vielä sovelluksia, jotka hyödyntävät tätä dataa nyt vielä arvaamattomalla tavalla.

LÄHTEET

1. IOT Finland 2015. Arrow ECS. Saatavissa: http://iotfinland.fi/. Hakupäivä 14.5.2017.

2. Haikala, Ilkka – Järvinen, Hannu-Matti 2003.Käyttöjärjestelmät. Helsinki: Talentum.

3. Karl, Holger – Willig, Andreas 2005. Protocols and Architectures for Wireless Sensor Networks. John Wiley & Sons Ltd.

4. Li, Suet-Fei – Sutton, Roy – Rabaey, Jan 2001. Low Power Operating System for Heterogeneous Wireless Communication Systems. Saatavissa: http://recerca.ac.upc.es/conferencies/PACT01/colp/paper01.pdf. Hakupäivä 19.5.2017

5. Shodhan, Sneha 2013. TinyOS. Saatavissa: https://www.slideshare.net/snecute/tinyos. Hakupäivä 1.5.2017.

6. TinyOS Documentation Wiki 2013. Saatavissa: http://tinyos.stanford.edu/tinyos-wiki/index.php/Main_Page. Hakupäivä 1.5.2017.

7. FreeRTOS FAQ - What is This All About? 2016. Real Time Engineers Ltd. Saatavissa: http://www.freertos.org/FAQWhat.html. Hakupäivä 4.5.2017.

8. Micrium Online Documentation 2017. Silicon Labs. Saatavissa: https://www.micrium.com/support/documentation/. Hakupäivä 2.5.2017.

9. Dunkels, Adam 2009. Contiki: Bringing IP to Sensor Networks. ERCIM News. Saatavissa: https://ercim-news.ercim.eu/en76/rd/contiki-bringing-ip-to-sensor-networks. Hakupäivä 2.5.2017.

10. Contiki: The Open Source OS for the Internet of Things. Thingsquare. Saatavissa: http://www.contiki-os.org/#why. Hakupäivä 14.5.2017.

11. What is the .NET Micro Framework? 2014. Microsoft. Saatavissa: http://www.netmf.com/what-is-netmf/. Hakupäivä 2.5.2017.

12. Huawei LiteOS 2015. Huawei Technologies Co., Ltd. Saatavissa: http://www.huawei.com/minisite/iot/en/liteos.html. Hakupäivä 2.5.2017.

13. Amadeo, Ron 2016. Google's new "Android Things" OS hopes to solve awful IoT security. Ars Technica. Saatavissa: https://arstechnica.com/gadgets/2016/12/google-brillo-rebrands-as-android-things-googles-internet-of-things-os/. Hakupäivä 2.5.2017.

14. Android Things. Google Developers. Saatavissa: https://developer.android.com/things/index.html. Hakupäivä 2.5.2017.

15. Nano-RK: A Wireless Sensor Networking Real-Time Operating System 2012. Jean-Philippe Lang. Saatavissa: http://www.nanork.org/projects/nanork/wiki. Hakupäivä 2.5.2017.

16. RIOT: The friendly Operating System for the Internet of Things 2017. FU Berlin. Saatavissa: http://www.riot-os.org/#usage. Hakupäivä 3.5.2017.

17. Learn about Windows 10 IoT Core 2017. Microsoft. Saatavissa: https://developer.microsoft.com/fi-fi/windows/iot/Explore/IoTCore. Hakupäivä 2.5.2017.

18. mbed OS 2017. ARM Ltd. Saatavissa: https://www.mbed.com/en/platform/mbed-os/. Hakupäivä 14.5.2017.

19. Legare, Christian 2014. Use A Kernel In Your IoT Designs. Electronic Design. Saatavissa: http://www.electronicdesign.com/embedded/use-kernel-your-iot-designs. Hakupäivä 4.5.2017.

20. Brown, Eric 2017. Shrinking the Linux Kernel and File System for IoT. The Linux Foundation. Saatavissa: https://www.linux.com/news/event/open-source-summit-na/2017/4/shrinking-linux-kernel-and-file-system-iot. Hakupäivä 4.5.2017.

21. Yocto Project 2016. Linux Foundation. Saatavissa: https://www.yoctoproject.org/about. Hakupäivä 4.5.2017.

22. Ubuntu Core 2017. Canonical Ltd. Saatavissa: https://www.ubuntu.com/core. Hakupäivä 4.5.2017.

23. Commodore 64 Connects To The Internet! (How to), 2014. Kooky Tech. Saatavissa: http://www.kookytech.net/2014/04/commodore-64-connects-to-internet-how-to.html. Hakupäivä 19.5.2017.

24. Connect the Dots 2017. Microsoft. Saatavissa: https://github.com/Azure/connectthedots. Hakupäivä 11.5.2017.

25. Comparison of Azure IoT Hub and Azure Event Hubs 2017. Microsoft. Saatavissa: https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-compare-event-hubs. Hakupäivä 11.5.2017.

26. What Is OceanConnect? 2017. Huawei Technologies Co., Ltd. Saatavissa: http://developer.huawei.com/ict/en/site-oceanconnect/article/ocean-connect-overview. Hakupäivä 11.5.2017.

27. What Is Weave? 2017. Google Developers. Saatavissa: https://developers.google.com/weave/guides/overview/what-is-weave. Hakupäivä 11.5.2017.

28. Overview of Internet of Things 2017. Google Developers. Saatavissa: https://cloud.google.com/solutions/iot-overview. Hakupäivä 12.5.2017.

29. Mitä on avoin data? 2010. Helsinki Region Infoshare. Saatavissa: http://www.hri.fi/fi/mita-on-avoin-data/. Hakupäivä 12.5.2017.

30. 6Aika-strategia 2016. Oulun kaupunki. Saatavissa: https://www.ouka.fi/oulu/kehittamishankkeet/6aika. Hakupäivä 12.5.2017.

31. Avoin data | Oulunliikenne.fi. 2017. Infotripla Oy. Saatavissa: http://wp.oulunliikenne.fi/wordpress/avoin-data/. Hakupäivä 12.5.2017.

32. Digitraffic rautatieliikenne 2015. Väestörekisterikeskus / avoindata.fi -palvelu. Saatavissa: https://www.avoindata.fi/data/fi/dataset/digitraffic-rautatieliikenne. Hakupäivä 12.5.2017.

33. Digitraffic tieliikenne 2016. Väestörekisterikeskus / avoindata.fi -palvelu. Saatavissa: https://www.avoindata.fi/data/fi/dataset/digitraffic-tieliikenne. Hakupäivä 12.5.2017.

34. Reaaliaikaista liikenne-valodataa auki Tampereella 2017. 6Aika. Saatavissa: https://6aika.fi/reaaliaikaista-liikennevalodataa-auki-tampereella/. Hakupäivä 12.5.2017.

35. Avatut ja avattavat tietoaineistot 2017. Ilmatieteen laitos. Saatavissa: http://ilmatieteenlaitos.fi/avoin-data-avattavat-aineistot. Hakupäivä 12.5.2017.

36. Ilmatieteen laitoksen avointa dataa jaossa Amazonin pilvipalvelussa 2017. Ilmatieteen laitos. Saatavissa: http://ilmatieteenlaitos.fi/tiedote/355219278. Hakupäivä 12.5.2017.